U.S. Pat. Appln. 09/830,462

SESSION ANNOUNCEMENT FOR ADAPTIVE COMPONENT CONFIGURATION

The present invention relates to the announcement of media sessions over a communications network and the adaptive configuration of application programs and components for receiving media sessions.

Multicast transmissions are becoming increasingly common on the Internet. In contrast to standard Internet Protocol (IP) point to point transmissions (unicast), IP multicast allows the simultaneous transmission of information to a group of recipients from a single source. Routing support for IP multicast transmissions is provided by the MBone (IP Multicast Backbone) which is a virtual network layered on top of the Internet.

IP multicast allows real-time communications over wide area IP networks and typical transmissions include video and audio conferencing, live multimedia training, university lectures and transmission of live television programmes.

A multicast transmission usually consists of a multimedia session made up of several individual media streams typically carrying video, audio, whiteboard or raw data. Some sessions are persistent, but the majority exist for a specific period of time, although need not be continuous. Multicast based transmissions on the MBone differ from unicast IP transmissions in that any user receiving the transmission can join the session (unless the transmission is encrypted) and to receive a transmission, a user need only know the appropriate transmission address and timing information.

Prior to a multicast transmission an appropriate announcement containing a session description is made, usually at an IP group multi-cast address. Standard session descriptions are generated using a Session Description Protocol (SDP), as defined in the Internet Engineering Task Force's draft RFC 2327. SDP is a simple ASCII text based protocol that is used to describe real time multimedia sessions and their related scheduling information. SDP messages are wrapped in a carrier protocol, known as a

Session Announcement Protocol (SAP), which, in addition to containing the necessary IP addressing and routing information for transmission across the Internet or MBone, allows the SDP message to be encrypted, signed or compressed. An announcement can then be sent at regular intervals to the announcement group address. As an alternative to SAP, a session may be announced by placing an SDP message on a World Wide Web site (WWW) or by sending it to individuals by email or as a unicast transmission inviting them to participate.

An SDP message conveys information about each media stream in the multicast multimedia session to allow the recipients to participate in the session. A typical SDP message will include the session name and purpose, the time(s) and date(s) the session will be active, the component media streams of the session and information required to participate in each media stream (IP multicast address, port, media format). The SDP message may also include details of the session's bandwidth requirements, an encryption key necessary to participate in a secure multicast transmission using public key encryption, contact information for the organiser of the multicast session, and a Unique Resource Indicator (URI) pointing to a WWW or an Intranet web site where further information on the session may be found, for example, background information relating to the conference.

The level of participation a user may make in a session or stream depends on its purpose. In a multicast television session, typically users would only be able to receive the session streams whilst in a multicast conference session the communication would be bi-directional with a central server (such as group address 120) receiving each participants transmissions and relaying them to the other participants. The level of participation expected of a user in a session or stream may be explicitly stated in the session description or it may be inherent from the session description, for example when a receive-only application is associated with a media stream type in the session

description.

A common front-end interface used by multicast end users is known as Session Directory Rendezvous (SDR). This interface takes the received announcements, decodes the SDP message and displays the names of those sessions that are still current in a list. The end user may then select one of the listed announcements to view further technical and user-oriented details of the announced session. From the displayed information, the end user can then select to join individual streams of the session or to join the entire session. Once the streams to be joined are selected, SDR starts the necessary multicast-enabled multimedia application on the end user's computer, such as Vic, Vat, or wb etc and passes the relevant stream information (a transport port address) from the announcement onto the application allowing the application to establish the link to the associated IP multicast address and participate in the stream at transmission time. Having initiated the applications and passed the relevant transport port address SDR plays no further part in the session.

Recent increased usage and demand for (multi)media sessions has highlighted a number of limitations in SDP. SDP limits session descriptions to defining a session having a single set of timings that apply to all of the streams within it. A session in which a stream starts mid-way through the transmission cannot easily be described using SDP. The structure of a session description written in SDP must be a simple linear list of streams which may not reflect the intuitive structure of a complex session. SDP supports a limited and predefined set of applications which can receive the streams and a limited and predefined set of transport mechanisms (e.g. Simple layering, RTP and UDP). As guaranteed Quality of Service (QoS) is becoming more and more desirable to the consumer and the supplier, the need to define QoS policies for the entire session and individual streams in terms of required system resources, bandwidth requirements and supported applications also needs to be met. There may also be requirements on

the prioritisation of streams and subsessions or more complicated rules about receiving streams. A further requirement on the part of the supplier is the need for charging facilities permitting the charging of an end user for a multicast transmission to which they subscribe according to the QoS and types of streams received etc. There is little scope to include information about QoS policies or charging within the conventional structure of an SDP session description, or any metadata about the session.

A further problem faced by providers of current (multi)media sessions and the developers of the associated (multi)media applications is the spread of skills required to implement an application that can initiate and manage a real-time data connection over a communications network and perform the (multi)media functions the end user would expect. For example, developers of multimedia applications require teams with skills in audio and video coding, network transport protocols, real time programming, user interface design and integration techniques. Furthermore, until now the only way a QoS policy could be implemented was to process a session description to determine which streams of a session could or should be run and then to initialise the applications so they connect to the respective streams. This required the communications manager not only to know about the session requirements and available system resources but also the capabilities of each application.

A further problem is that applications should be able to adapt to available network and host resources. This is particularly important for multi-party applications operating in heterogeneous environments where each party may have different resources available to them. Furthermore the nature of the heterogeneity may vary over the lifetime of the session, for example as network congestion varies or as the terminal resources are shared with other applications or other users. The present invention is able to use a QoS policy incorporated within the session description to prioritise the allocation of resources and to determine whether participation in the session is viable.

Another problem is that the application developer and service provider typically need to address security and charging requirements. The present invention allows security and charging policies to be incorporated within the session description for use within the session control system to invoke appropriate charging and security procedures. Instead of having to develop security and charging functions the application developer and service provider need only specify appropriate policies. These polices may relate to remote services identified in the session description and invoked by the session control system once the user has elected to receive a particular media session.

BRIEF SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided a method of announcing a description of a media session, the method comprising the steps of:-

generating a session description comprising media oriented data necessary for a user to receive at least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session; and,

announcing the media session by making the session description available to potential recipients of the media session.

Preferably, said media oriented data prescribes a number of application program components to be used in order to build an application to participate in the media session.

Conveniently, the media oriented data prescribes a manner in which the components are to be configured to build the application program

In preferred embodiments, the session description is generated using a structured data format.

Preferably, the structured data format conforms to the format of Extensible Mark-Up Language.

Conveniently, the user oriented data for the or each media stream is generated in a one or more respective media modules within the session description, and said method comprises the further steps of:-

generating a base module comprising user oriented data relevant to the media session;

providing a link between the base module and the or each media module;

announcing the media session by making the base module available to said potential recipients;

wherein the link to the or each media module permits a user to access the or each media module subsequent to the base module.

According to a second aspect of the invention there is provided a method of configuring a platform for receiving a media session, said method comprising the steps of:-

receiving a session description of a media session, said session description comprising media oriented data necessary for a platform to receive at least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session;

processing said session description to determine an appropriate application program configuration form a list of available application programs or program components;

configuring a respective media session application program from said list of available programs for participation in said media session.

The method may further comprise the steps of receiving network data relating to characteristics of the network over which said media session is to be transmitted and

wherein the respective media session application is configured according so said network data.

The method may further comprise the steps of receiving terminal data relating to characteristics of the terminal on which said media session is to be received and wherein the respective media session application is configured according so said terminal data.

Preferably, said network data or terminal data or both is monitored during the media session and the media session description is modified in response to changes to the monitored data.

In preferred embodiments, the method further comprises the steps of receiving user profile data relating to preferences of a user of the media session and wherein the respective media session application is configured according so said user profile data.

Conveniently, the session description may further comprise data defining a quality of service policy for receiving the media session and the respective media session application is configured according to said quality of service policy.

The session description may further comprise data defining one or more remote services necessary for participation in said media session and the respective media session application is configured according to requirements of said one or more remote services.

Preferably, the step of processing the session description comprises the step of parsing the session description using a terminal session control to determine an appropriate application program configuration form a list of available application programs or program components; selecting one or more media streams identified in the session description; and connecting the or each selected media stream to one or more application programs or components in said configuration by means of a session control configured for managing media stream connections for the or each application program

or component.

According to a third aspect of the invention there is provided a system for announcing a description of a media session, the system comprising:-

a media session description generator for generating a session description comprising media oriented data necessary for a user to receive at least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session; and,

a media session announcer for announcing the media session by making the session description available to potential recipients of the media session.

According to a fourth aspect of the invention there is provided a system for configuring a platform for receiving a media session, said system comprising:-

a receiver for receiving a session description of a media session, said session description comprising media oriented data necessary for a platform to receive at least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session;

a processor for processing said session description to determine an appropriate application program configuration form a list of available application programs or program components; and,

an application builder for configuring a respective media session application program from said list of available programs for participation in said media session.

In the above aspects of the invention application program development is simplified by using the session description to drive the dynamic management and

configuration of the client system on which they are implemented and to adapt the client system according to available network and terminal resources. It also reduces the problem of handling charging and security requirements to a matter of specifying charging and security policies within the session description which can then be implemented using services provided by a remote server .

The present invention is particularly useful when used in conjunction with the modular session description described in this patent application and which is also the subject of our co-pending UK patent application 9826158.9, and when used in conjunction with the session control system also described in this patent application and the subject of our co-pending UK patent application 9826157.1.

An example of the present invention will now be described in detail with reference to the accompanying drawings, in which:

Figure 1 is a schematic diagram illustrating a multicast transmission across the MBone;

Figure 2 is a schematic diagram illustrating the distribution of an SDP announcement;

Figure 3 is a block diagram of a modular session description of a simple session generated in accordance with the present invention;

Figure 4 is a block diagram of a modular session description of a complex session generated in accordance with the present invention;

Figure 5 is a schematic diagram of a system for managing media stream connections;

Figure 6 is a flow chart illustrating the steps involved in managing a media session according to the system of Figure 5; and,

Figure 7 is a flow chart further illustrating a parsing step of Figure 6.

Figure 8 is a schematic diagram of a Graphic User Interface (GUI) defined in a

session description according to an aspect of the present invention.

Figure 9 is a schematic diagram of a further system for managing media stream connections at a terminal;

Figure 10 is a schematic diagram of a client-server environment including a client application builder; and,

Figure 11 is a schematic diagram showing the client application builder of Figure 10 in greater detail.

An example of an IP multicast transmission system is described with reference to Figure 1. Prior to a multicast transmission, an appropriate announcement containing a session description is made, thereby allowing end users 110a-110e to elect to receive the transmission. Each end user electing to receive the transmission is linked to a group IP Multicast address 120 associated with the transmission. At the transmission time of the multicast session, the session streams are transmitted from a source 130, or a plurality of sources, to the group address. At the group address, the transmission is disseminated along the links 140 to those end users who have elected to receive it (in this example end users 110a-110c).

An example of an announcement and election system is described with reference to Figure 2. Most public multicast sessions are announced at a single group IP multicast address 200 dedicated to the transmission of announcements to multicast end users. End users 210a-210e electing to receive the announcements are linked to the announcement group address and, in the same way as an actual session transmission, each announcement arriving at the announcement group address is disseminated to the end users. A front-end interface 220 on each end user's computer displays information obtained from the associated session description for each announcement. The minimum information a session description may contain is a time and date that the session will be active and the group IP multicast address(es) from

which the end user may elect to receive one or more media streams and to which they could send their own streams for the session. Using the front-end interface, an end user can select the announced session(s), or their component stream(s) they wish to participate in.

Figure 3 is a block diagram of a session description 300 for a simple multicast television session. The session description 300 comprises a base module 310 linked to a media module 320.

The base module 310 contains user-oriented data relating to the session including the title and timing information. The base module 310 may also include a description or abstract, contact information about the organiser and a WWW or an intranet URI pointing to a web site containing further information. Ideally, the base module 310 should contain enough information for the user to decide if they are interested in participating in the session.

The media module 320 contains announcement data relating to a video stream of the session. The media module 320 contains the technical information (data) necessary for the user to receive the associated media stream. In particular, connection, timing and media format details are provided.

A first example of a session description 300 generated for transmission to end users is shown below:

```
(
type=(base)
id=(310)
info=(title="live multicast television session")
source=(name="A.Sender" email=asender@tx.com)
media=(video=(client=odbits0.16))
time=(length=50m repeat=continuous)
category=("Entertainment")
options=(none)
modules=(m=320)
)
```

```
(
type=(media)
id=(320 310)
media=(video=(client=odbits0.16))
connection=(229.1.1.2/7000)
time=(length=50m)
)
```

Session description example 1

The base module 310 has a unique identifier (id field) used in the generation of links between two modules during the processing of the session description. The module field of the base module 310 lists the type and unique identifier of the media module 320 linked to the base module 310. The second identifier in the id field of the media module 320 is the unique identifier belonging to the base module 310 linking the media module back to the base module 310. By extension, these two-way links permit a module tree to be traversed from a base module downwards or from a media module upwards. The use of this feature is described later with reference to session description example 4.

The connection field of the media module 320 contains the IP multicast address and port number from which the media stream can be received.

Figure 4 is a block diagram of a session description 400 for a complex multicast session of a multimedia conference with two tracks, or sub-sessions, and a panel discussion. Each track provides multiparty video and audio conferencing and a shared whiteboard for leaving notes and messages. The panel discussion is encrypted and the whole conference is subject to a subscription fee payable in advance by each participant.

The session description 400 contains a top level base module 410 linked to further base modules 420, 430, 440 and an options module 411. The top level base module 410 contains data relating to the overall session including its name, purpose

and timing information. The options module 411 contains details of the payment mechanism for subscription fees.

Each further base module 420, 430, 440 relates to a subsession of the conference. Base module 420 relates to the first track of the conference. The base module 420 is linked to media modules 421-423, each containing connection, timing and media format data for respective video, audio and whiteboard streams.

The base module 420 is also linked to options module 424 which contains data relating to a QoS policy for the first track defining which media modules are optional and which are mandatory for a participant of the first track. The mandatory list contains identifiers of those media modules which are needed for the session or subsession to operate correctly whilst the optional list contains identifiers of the media modules that are not necessary for the session or subsession to operate correctly if system resources are scarce.

The base module 430 relates to the second track of the conference. It is linked to media modules 431-433, each containing connection, timing and media format details for respective video, audio and whiteboard streams. The base module 430 is also linked to options module 434 which contains data relating to a QoS policy for the second track defining which media modules are optional and which are mandatory for a participant of the second track. Base module 440 relates to the panel discussion. It is linked to media modules 441 and 442, each containing connection, timing and media format details for respective video and audio streams of the panel discussion. The base module 440 is also linked to options module 443 which contains encryption details (ie. how and where to get the necessary cryptographic keys) necessary for a participant to decode the panel discussion media streams 441, 442 according to a known encryption mechanism such as DES or public key encryption.

The video media stream defined in media module 441 is layered. Layering of

media streams allows users with different system resources to receive as much of the stream as their system resources allow. Every user must receive the bottom layer of the stream containing the minimum stream data. However, if a user has sufficient free system resources they can receive the next layer up containing enhancements to the previous layer. Successive layers can be received enhancing the received media stream until the maximum number of layers is received or all free system resources capacity is used. The media module 441 is linked to an options module 444 which contains data on the layering necessary for the end user to be able to receive the layered stream correctly.

The portion of the session description 400 generated for modules 410, 411, 420 and 440 for transmission to end users is shown below in session description example 2.

```
( # overall conference session
  type=(base)
  id=(410)
  info=(title="Multimedia98 Conference")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(client=RealPlayerG2) whiteboard=(client=wb))
  time(start="09:00 GMT 25/12/98" stop="13:00 GMT 25/12/98")
  options=(oc=411)
  modules=(b=420 b=430 b=440 oc=411)
)

( # conference track 1
  type=(base)
  id=(420 410)
  info=(title="MM98 Systems and Applications Track")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(client=RealPlayerG2) whiteboard=(client=wb))
  time(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
  options=(osq=424)
  modules=(m=421 m=422 m=423 osq=424)
)

( # session QoS for track 1
  type=(option-sQoS)
  id=(424 420)
  mandatory=(421 422)
  optional=(423)
)
```

```
( # conference panel discussion
  type=(base)
  id=(440 410)
  info=(title="MM98 Panel Discussion")
  source=(name="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(client=RealPlayerG2) whiteboard=(client=wb))
  time(start="11:00 GMT 25/12/98" stop="13:00 GMT 25/12/98")
  options=(osec=443)
  modules=(m=441 m=442 osec=443)
)

( # video for panel discussion
  type=(media)
  id=(441 440)
  info=(title="MM98 Panel Discussion Video")
  source=(owner="Joe Bloggs" email=joe@nowhere.com)
  media=(video=(type=live client=RealPlayerG2))
  connection=(226.0.0.106/1010 policy=444)
  time=(start="11:00 GMT 25/12/98" stop="13:00 GMT 25/12/98")
)

( # media QoS policy for panel discussion video
  type=(option-mQoS)
  id=(444 440)
  mechanism=(layer=(base=226.0.0.106/1010 number=3))
)

( # encryption policy for panel discussion
  type=(option-sec)
  id=(443 440)
  participant=(member=w3c)
  publickey=(location=http://www.w3.org/members_only/)
  info=(location=http://www.w3.org/)
)

( # charging policy for entire conference
  type=(option-chg)
  id=(411 410)
  mechanism=(type=AAA)
  price=(fee=1000GBP)
  info=(location=http://www.aaa.net/)
)
```

Session description example 2

Where there is surplus network bandwidth available, complete session descriptions can be announced to end users who may then elect to receive the announced session or parts thereof. However, the individual modules of the session

description do not need to be announced together. If the network bandwidth available for announcements restricts the size of session descriptions, only the top level base module may be announced. In this situation, the link between modules may be, for example, a URI to a WWW or an intranet web site or server, an email address, an IP multicast address, an FTP address or details of a file or database stored on a local computer system from which an interested user can obtain the remaining modules.

The following session description example illustrates how the above session description for base module 420 would be changed if media module 421 were stored on a WWW server:

```
(  # conference track 1
   type=(base)
   id=(420 410)
   info=(title="MM98 Systems and Applications Track")
   source=(owner="Joe Bloggs" email=joe@nowhere.com)
   media=(video=(client=RealPlayerG2) whiteboard=(client=wb))
   time(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
   options=(osq=424)
   modules=(m=421  location=http://www.announce.org/cgi-bin/module.cgi?id=421
            m=421 m=423 osq=424)
)
```

Session description example 3

Furthermore, top level modules of a session description may be announced well in advance of the actual transmission, at a time where the final details of content are unknown, in which case the remaining levels may be made available from pre-announced links at a later time.

Figure 5 is a schematic diagram of a system for managing media stream connections at a terminal of an end user system according to the present invention.

The session control system 500 is linked to an announcement receiving interface 510 and one or more multicast-capable multimedia applications 520. The session control system 500 and the announcement receiving interface 510 are connected to a

network interface 530 via which announcements may be received and multicast transmissions may be initiated and/or received.

Announcements received at the network interface 530 are routed to the receiving interface 510. The receiving interface 510 decodes each announcement to obtain the session description and displays the user oriented information from the one or more base modules in a list to the user. The user is able to select a session description from the list announcing a session they wish to receive. The selected description is passed to the session control system 500 which determines which of the user's multimedia applications 520 are required for participation in the described session, starts the applications and initiates and provides the necessary media streams to the respective applications 520 via a communications manager 550.

The receiving interface 510 may be linked to other Internet communications applications 540 such as a WWW browser or an email client (not shown) which may be used to gather further information on the described session based on links provided in the session description. Also, where an incomplete set of base and/or media modules of a session description are received, the receiving interface 510 attempts to obtain the remaining modules using the Internet communications applications prior to passing it onto the session control system 500.

Figure 6 is a flow chart showing the steps taken by the session control system 500 upon receipt of a session description. The description is first parsed in step 600 to identify client applications for each media module. Once this is done a second parse is carried out where applications are launched in step 610, that is to say for each media module start the application specified in the client field if that application has not already been started. The portion of the session description relating to the respective media type, i.e. the media module, the base module directly above the media module, all other modules attached to that base module and any other options modules that

apply, is passed to the corresponding application in step 620. Since the media modules are marked with appropriate client applications, each application will be able to select those media streams that it wants to participate in. The application replies to the session control system with a connection request specifying its requirements in the form of a list of identifiers of media modules from which streams are to be initiated in step 630. The connection request is assembled by the session control system in step 640 and the system then parses the session description to identify other applications to launch in step 645. If a further media type is found, steps 610 to 640 are repeated, otherwise the session control system uses the assembled connection requests to form a list of media modules. This list is passed, together with a session QoS policy, to the communications manager, a system used in by the session control system, which determines according to the QoS policies and available system resources whether each connection request is viable.

The session QoS policy is constructed in two steps:- first, the multiple session QoS policies relevant for all the media modules to be initiated are combined into one session QoS policy: second, the resulting session QoS policy may be adapted to take account of (a) user default preferences (defined in a user profile), (b) a user's wish to determine the policy interactively, and (c) an application's default configuration (defined in the application profile(s)).

The communications manager responds to the session control system in step 650 with an indication of the viable media stream connection requests. If necessary, the session control system may contact a charging system to initiate accounting for the session prior to requesting the communications manager to create the viable media stream connections in step 660.

Once a session starts, each received data stream relating to the session is passed to the associated multimedia application in step 670 until the scheduled stream

time ends in step 680 or the multimedia application requests to the session control system that the connection is terminated in step 690, at which point the session control system disconnects the connection in step 700.

Figure 7 is a flow chart showing the QoS management step 650 of Figure 6 in greater detail.

Having received the assembled list of connection requests, the communications manager matches each item of this list to a media profile in step 705. A media profile defines requirements which must be met for the requested media stream to operate on the end user's computer including the minimum network bandwidth needed for satisfactory reception of the stream.

A terminal profile is determined in step 710. The terminal profile defines the resources which are available at the end user's computer for use by the requested media streams. This includes available network bandwidth, free memory and disk space and available hardware such as monitor size, processor speed and free audio and video capture devices. The media profile of each connection request is compared against the available system resources defined by the terminal profile in step 720. If the terminal profile matches or exceeds the media profile, the connection request is declared viable in step 730 and the terminal profile is decremented accordingly for the remaining connection requests in step 740. Each connection request is processed until there are no remaining requests or until the media profile of a request exceeds the terminal profile. In this situation, the communications manager determines the optimum terminal profile the user's computer would have if all non-essential applications were not running in step 750 and whether the computer is capable of fulfilling the media profile in step 760. If the computer is capable of fulfilling the media profile, the communications manager attempts to free system resources from currently allocated streams or connection requests which have lower priority or by asking the user to terminate other

non-essential applications running on the computer in step 770. Alternatively, this could be done by reducing the number of layers received from a layered stream transmission. If sufficient resources cannot be found an exception is reported to the user and the connection request is marked as unviable. If the media stream that cannot be received is defined as mandatory in a QoS policy for a media session or subsession, all the connection requests for that media session or subsession are cancelled in step 790. If, however, the media stream is optional, the communications manager continues processing further connection requests in step 720. Once all pending connection requests have been processed, the communications manager reports those that are viable to the session control system.

The processing of a session description will now be described with reference to Figure 4 and session description example 4 which is the session description generated for Track 1 (modules 410 and 420-424 of Figure 4).

```
(  # overall conference session
   type=(base)
   id=(410)
   info=(title="Multimedia98 Conference")
   source=(owner="Joe Bloggs" email=joe@nowhere.com)
   media=(video=(client=RealPlayerG2) whiteboard=(client=wb))
   time(start="09:00 GMT 25/12/98" stop="13:00 GMT 25/12/98")
   options=(oc=0010)
   modules=(b=420 b=430 b=440 oc=411)
)

(  # conference track 1
   type=(base)
   id=(420 410)
   info=(title="MM98 Systems and Applications Track")
   source=(owner="Joe Bloggs" email=joe@nowhere.com)
   media=(video=(client=RealPlayerG2) whiteboard=(client=wb))
   time(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
   options=(osq=424)
   modules=(m=421 m=422 m=423 osq=424)
)

(  # video for track 1
   type=(media)
   id=(421 420)
```

```
    info=(title="MM98 Systems and Applications Track Video")
    source=(owner="Joe Bloggs" email=joe@nowhere.com)
    media=(video=(type=live client=RealPlayerG2))
    connection=(226.0.0.100/1000)
    time=(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
)

( # audio for track 1
    type=(media)
    id=(422 420)
    info=(title="MM98 Systems and Applications Track Audio")
    source=(owner="Joe Bloggs" email=joe@nowhere.com)
    media=(audio=(type=live format=g711))
    connection=(226.0.0.101/1001)
    time=(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
)

( # whiteboard for track 1
    type=(media)
    id=(423 420)
    info=(title="MM98 Systems and Applications Track Whiteboard")
    source=(owner="Joe Bloggs" email=joe@nowhere.com)
    media=(whiteboard=(client=wb))
    connection=(226.0.0.102/1002)
    time=(start="09:00 GMT 25/12/98" stop="11:00 GMT 25/12/98")
)

( # session QoS for track 1
    type=(option-sQoS)
    id=(424 420)
    mandatory=(421 422)
    optional=(423)
)
```

Session description example 4

The session control system, having received the above session description, processes the tree structure of the session description starting at base module 410. The first module encountered is base module 420. As this is not a media module but it does have sub-modules, the session control system continues down this branch to media module.

The media field of the media module 421 already defines the multimedia client application required as RealPlayerG2 (a multimedia application of Real Networks Inc)

thus the session control system ignores it and continues to the next media module. The media field of the media module 422 does not have a multimedia client application defined, however a format for the audio data is specified. The session control system recognises that this particular audio format can be supported by RealPlayerG2 so it amends the media field to read client = RealPlayerG2. The next media module 423 has already defined a client application as wb so it ignores this module, and it also ignores the option module 424.

The session control system parses the tree structure again in order to launch client applications. The first media module 421 specifies that RealPlayerG2 should be launched, hence the session control system launches the application on the end user's system and keeps a record of this activity. The second media module 422 specifies an application that has already been launched and so the session control system ignores it and continues to the next media module. The media module 423 specifies that wb should be launched, so the session control system launches the application and keeps a record of this activity.

RealPlayerG2 is passed the media module 421, base module 420 and modules 422-424. The application processes the media modules given to determine which it can handle, and in this case it identifies 421 and 422. Having determined which streams it can handle, the application sends a connection request back to the session control system requesting connection to the media streams of modules 421 and 422. Similarly, wb is passed the media module 423, base module 420, modules 421-422, and the module 424. The application processes the given modules as described previously, and requests connection to the media stream of modules 423.

The above connection requests are assembled by the session control system into a list, this list is then passed to the communications manager along with the session QoS policy module 424. The communications manager determines whether

each request is viable according to the steps of Figure 7.

Assuming there are sufficient resources for all the connection requests for mandatory media streams, the communications manager passes back a list of viable streams to the session control system which then processes the tree again to determine the connection data held in the connection field of each media module so it can request that the communications manager initiate a connection to the appropriate media stream for each of the viable connection requests according to the connection data. The session control system then manages the session and its media stream connections as is described with reference to steps 670 to 700 of Figure 6.

The foregoing examples of the present invention have mostly been concerned with media session descriptions in which the client application required to receive a constituent media stream is explicitly prescribed. Session description example 2 discussed above is an example of a session description in which the client application is explicitly prescribed, for example "client = RealPlayerG2". If the client application is capable of accepting more than one media stream encoding format then this may also be prescribed in the session description. Additionally or alternatively the session description may prescribe a blueprint of client application program components including the required configuration of the components, such as interfacing and data passing, to be instantiated at the user's terminal for participation in a media session. The components may be, for example, for media transmission/reception, encryption or compression.

An example of a media module for such an approach is shown below:

(# Java Bean client blueprint declaration for Text chat session
type = (media)
id=(1030,1000)
info=(title="Text Chat")

```
source=(owner="Joe Bloggs" email=joe@nowhere.com)
media=(Java Bean=(
        <bean class="java.util.Vector">
         <add>
          <bean class="nowhere.com.chat.StringMulticastBean" id="Multicaster">
           <property name="debug">
            <cast class="boolean"><string>false</string></cast>
           </property>
           <event-binding name="message">
            <script>
             <bean source="TextArea">
              <call-method name="append">
               <property target="event:arg1" name="message"/>
              </call-method>
              <call-method name="append">
               <string>&#10;</string>
              </call-method>
             </bean>
            </script>
           </event-binding>
          </bean>
         </add>
         <add>
          <bean class="java.awt.Frame">
           <property name="title" value="Chat"/>
           <add>
            <bean class="java.awt.TextArea" id="TextArea">
             <property name="editable">
                                  <cast class="boolean">
              <string>false</string>
             </cast>
            </property>
           </bean>
           <string>Center</string>
          </add>
          <add>
           <bean class="java.awt.Panel" id="Panel">
            <add>
             <bean class="java.awt.Button">
              <property name="label" value="Clear"/>
              <event-binding name="action">
               <script>
                <property target="TextArea" name="text">
                 <string/>
                </property>
               </script>
```

```
                </event-binding>
               </bean>
              </add>
             </bean>
             <string>East</string>
            </add>
            <add>
             <bean class="java.awt.Panel">
              <add>
               <bean class="java.awt.TextField" id="TextField">
                <args>
                 <cast class="int"><string>20</string></cast>
                </args>
               </bean>
              </add>
              <add>
               <bean class="java.awt.Button">
                <property name="label" value="Send"/>
                <event-binding name="action">
                 <script>
                  <bean source="Multicaster">
                   <call-method name="send">
                    <property target="TextField" name="text"/>
                   </call-method>
                  </bean>
                  <property target="TextField" name="text">
                   <string/>
                  </property>
                 </script>
                </event-binding>
               </bean>
              </add>
             </bean>
             <string>South</string>
            </add>
           </bean>
          </add>
         </bean>)
    connection=(524.0.0.100/12345)
)
```

Session Description example 5

In this example, the media field prescribes a number of application components

and the manner in which they are to be configured to support user participation in a media session. The media field of example 5 is defined using Bean Mark Up Language (BML), a scripting language developed by IBM for describing structures of nested and interconnected JavaBeans components. JavaBeans is a standard developed by Sun Microsystems for defining software components. JavaBeans components or Beans are reusable software components that are written in the JAVA programming language. JavaBeans components can be combined together to create executable programs. BML is directly executable, that is to say, processing a BML script such as the media field in example 5 results in an application program configured according to the script. BML has elements that can be used to describe the creation of new JavaBeans, access existing JavaBeans, configure Beans by getting or setting properties or fields, binding so called "events" to other Beans and calling methods contained within Beans.

The BML script of example 5 comprises a script for describing a multicast "chat" application, that is to say, a program application allowing one or more users to exchange text based messages. Figure 8 shows a graphical user interface (GUI) that is configured and executed as a result of processing the BML script of Example 5. The script of Example 5 will now be explained with regard to the GUI application of Figure 8.

The BML script defines a Frame "Text Chat" which defines the rectangular periphery 800 of the GUI shown in Figure 8. The BML script further comprises:-

i) a "StringMulticastBean" JavaBeans component for the transmission and reception of text based message data;

ii) a non-editable "TextArea" JavaBeans component for displaying messages; The "TextArea" component defines the rectangular text area 805 in Figure 8 displaying the text lines:-

"Can you send me a screen shot of the chat application"

"What format would you like it in"

iii) a "Panel" component containing a "Button" component labeled "Clear" for clearing the text area 805; The Panel and Button components define the soft button "Clear" 810 to the right of the text area 805 in Figure 8.

iv) a "Panel" component containing a "TextField" component for displaying messages typed by the user of the GUI to be sent to other participants of the media session and a "Button" component labeled "Send" for sending the text in the "TextField" component. The "TextField" component defines the rectangular text area 815 positioned towards the bottom edge of the GUI periphery 800 in the drawing of Figure 8. The associated "Button" component defines the soft button 820 labeled "Send" to the right of the text area 815 in the drawing of Figure 8.

In example 5 the part of the script between the <event-binding name="message">.....</event-binding> tags, declares that any messages received from the network are passed to the "TextArea" component. The part of the script between the first <event-binding name="action">.....</event-binding> tags, declares that pressing the Clear button 810 on the GUI clears the text represented in the text area 805. In addition, the part of the script between the second <event-binding name="action">.....</event-binding> tags, declares that pressing the Send button 810 on the GUI passes the contents of the "TextField" component to the "StringMulticastBean" component for transmission.

Prior to being configured, the application described by the BML script in example 5 is passed further information from the information field and connection field of the session description to insatiate the application for the particular session. This information is kept separate in the different fields of the session description so as to separate the application description from the session specific parameters.

In the example 5 the BML defined application is configured at run-time from

available JavaBean components using a BML player or BML compiler. In another example the session description, or at least the media field, is adapted to be written in Extensible Markup Language (XML) and is transformed from XML to BML at run time using an appropriate Extensible Sytlesheet Language (XSL) transformation script.

As mentioned, in example 5 the media field prescribes a number of application components and the manner in which they are to be configured to support user participation in a media session. However, in another example, as an alternative to the above prescriptive approach, a media session description may declare the client's requirements for participation in the media stream and leave it to the end user's terminal to determine the client application program or programs or components that are required to participate in that media stream. In declaring the requirements, the session description may define the encoding format of the media stream and leave it to the end user's terminal to find a client application capable of handling that encoding format. Session description example 4 discussed above is an example of this. In another example a number of components types (e.g. video, audio, encryption etc) are declared instead and the user's terminal selects from the components available to it and instantiates an appropriate application for participation in the session. For example, media session properties such as reliability, cost and latency may be specified in a session description so that components can be selected according to specific requirements. For instance: reliable, low cost, high latency components may be more appropriate for media stream re-transmission; moderately reliable, moderate cost, moderate latency components may be more appropriate for supporting media stream data redundancy; and, high cost, highly reliable, low delay components may be more appropriate for supporting media streams or sessions that require a high quality of service. Selection of one or more appropriate components may be implemented by a knowledge-based system, for example.

An example of a session description for declaring a required media stream encoding format is shown below:

```
( # Video client declaration
  type = (media)
  id = (1010,1000)
  info = (title = "Video stream")
  source = (owner = "Joe Bloggs" email = joe@nowhere.com)
  media = (video = (codec = mpeg width = 600 height = 400 depth = colour))
  connection = (500.146.108.64)
  timing = (link 1020)
)

( # Audio client declaration
  type = (media)
  id = (1020,1000)
  info = (title = "CD Quality Audio Stream")
  source = (owner = "Joe Bloggs" email = joe@nowhere.com)
  media = (audio = (type = (codec = G711) timing = synchroniser ticker = stock_ticker))
  source = (500.146.107.25)
)
```

Session description example 6

As mentioned in relation to example 5, the session description, or at least the media field of the description, may be written in Extensible Markup Language (XML) and transformed from XML to BML at run time using an appropriate Extensible Sytlesheet Language (XSL) transformation script.

Figure 9 is a schematic diagram of another system for managing media stream connections at a terminal. The system operates in a manner that is similar to that which has been described with reference to Figures 5, 6 and 7. In particular, this example is particularly useful when a session description includes declarative or prescriptive components such as those described above.

The session control system 900 includes a session description parser 910, an application builder 920, a number of applications and application components 930 and a

communications manager 940. Upon receipt of a session description by the session control system 900, the session description parser 910 parses the session description to determine client application requirements for the media streams of the media session described by the session description. The client application requirements are passed to the application builder 920 which generates a number of possible application configurations based on the available applications and application components 930. In combination with the communications manager 940, the application builder evaluates each application configuration to determine the best configuration which is the closest match to the client application requirements whilst satisfying predetermined criteria such as quality of service, resource utilisation and user preferences. The best configuration, which may be an existing application instead of a combination of components, is then selected and instantiated. The components may include Java Beans, COM objects, MHEG objects, components from dynamic linked libraries or applications that are linkable to others such as the ActiveMovie application.

As an example, in a multimedia conference an application may be required that includes an audio component and an encryption component. Instead of selecting an application requiring many resources and offering facilities such as video which are not supported by the conference, individual Java Bean components are combined by the application builder to provide the application.

The application builder may be configured to determine the best application configuration in combination with the communications manager by heuristic methods or other artificial intelligence methods such as knowledge-based systems, neural networks or genetic programming.

Due to the heterogeneity of the Internet and differing capabilities and operating environments of end user computer systems, the session control system described has been implemented in Java (Java is a Trade Mark of Sun Microsystems Inc.). The

announcement receiving interface, Session Directory, receives the announcements and passes those selected by the end user to the session control manager implemented as an application programming interface running as a background process on the end user's computer.

An application builder suitable for use in the system of Figure 9 will now be described with reference to Figures 10 and 11. In Figure 10 a networked client 1000 in a client server environment comprises a application builder 1005, a session directory 1010 a network interface 1015 and a number of application components including an application GUI 1020, client applications/components of remote services 1025, an application control 1030 for controlling the remote services and a number of media stream communication channels 1035. The client 1000 is connected to network 1045 which comprises other clients 1050, media servers 1055 and servers of remote services 1060. In the context of the present example, a remote service is a service offered by a service provider at a location remote from the client 1000. In the present example remote services include:- ecommerce services such as payment services, authorisation services, certification services, key servers (for public key distribution), mail servers, directory services such as white and yellow pages, and charging services as may be included in an associated options module of the session description.

In Figure 11 the application builder 1005 comprises an application description writer 1105 and an application builder tool 1110. The application description writer comprises a first interface (not shown) for receiving a session description 1115 from the session directory 1010, a second interface (not shown) for receiving data defining a network profile 1120 describing various characteristics of the network over which media sessions are to be transmitted, a third interface (not shown) for receiving a data defining a user profile 1125 describing various preferences of an end user who is to receive a media session, and a fourth interface (not shown) for receiving data defining a

terminal profile 1130 describing various resource related characteristics of a terminal 1135 on which the media sessions are to be received. A fifth interface (not shown) is provided for receiving data defining a component profile 1140 of application components 1145 that are available to the application builder either locally or from the remote clients 1050 or servers 1055,1060.

In one example, the application description writer receives a session description 1115 in the form of an XML script from the session directory and at least a component profile 1140 also in the form of an XML script. The session description may be prescriptive or declarative or include both prescriptive and declarative elements. The description writer may additionally receive a network profile 1115 describing current network access constraints for example, a user profile 1125 describing specific user preferences, for example high quality audio in preference to low quality audio and video if the network is congested, and a terminal profile 1130 describing current terminal capabilities. The profiles are also expressed in XML script for ease of processing by the application description writer 1105. The application description writer comprises an XSL transformation script which processes the session description to determine an optimum component configuration based on the session description and the resource related profiles. The output of the application description writer is a BML script 1150. This script is processed by the builder tool 1110 to instantiate an application 1155 for use by the client 1000 in the selected media session.

Thus, in order to process the session description an XSL script is run which outputs a BML script to a temporary file. When, for example, a <chat> tag is parsed in the session description the BML script for the chat application is written out. Similarly, when an <address> tag is parsed, BML script to set the multicast address for the chat application is written out. The terminal, network and user profiles are also provided in XML and rules for modifying the parsed BML script based on the profiles are

represented in XSL. The BML script is modified for each profile by processing it with respect to each respective XSL script. For example, if a low bandwidth connection is provided for an audio-video conference the video media stream may be dropped on the basis of bandwidth information in the network profile.

In the following example a BML script for an audio-video conference is presented. After processing the network profile the video component is removed if bandwidth constraints identified in the network profile so determine. Under these circumstances all the script between the first <add> tag to the first </add> tag.

```
<?xml version="1.0"?>

<bean class="java.util.Vector">
 <add>
  <bean class="com.bt.visual.lsma.beanlava.VidMainBean" id="video">
     <property name="numLayers">
          <cast class="int"><string>3</string></cast>
     </property>
     <call-method name = "setMcastLayer">
       <cast class="int"><string>0</string></cast>
       <string>226.1.2.1</string>
       <cast class="int"><string>6006</string></cast>
    </call-method>
     <call-method name = "setMcastLayer">
       <cast class="int"><string>1</string></cast>
       <string>226.1.2.2</string>
       <cast class="int"><string>6008</string></cast>
    </call-method>
     <call-method name = "setMcastLayer">
       <cast class="int"><string>2</string></cast>
       <string>226.1.2.3</string>
       <cast class="int"><string>6010</string></cast>
    </call-method>
     <call-method name = "setMcastLayer">
       <cast class="int"><string>3</string></cast>
       <string>226.1.2.4</string>
       <cast class="int"><string>6012</string></cast>
    </call-method>
    <call-method name = "setGoing"/>
```

```
  </bean>
 </add>
 <add>
  <bean class="com.bt.visual.lsma.beanlava.AudMainBean" id="audio">
     <property name="numLayers">
          <cast class="int"><string>3</string></cast>
     </property>
     <call-method name = "setMcastLayer">
      <cast class="int"><string>0</string></cast>
      <string>226.4.5.4</string>
      <cast class="int"><string>4000</string></cast>
    </call-method>
     <call-method name = "setMcastLayer">
      <cast class="int"><string>1</string></cast>
      <string>226.4.5.5</string>
      <cast class="int"><string>4002</string></cast>
    </call-method>
     <call-method name = "setMcastLayer">
      <cast class="int"><string>2</string></cast>
      <string>226.4.5.6</string>
      <cast class="int"><string>4004</string></cast>
    </call-method>
    <call-method name = "setGoing"/>
  </bean>
 </add>
 <add>
  <bean class="java.awt.Frame" id="frame">
   <property name="title">
     <string>Audio Test</string>
   </property>
   <property name="background" value="0x1eeeee"/>
   <property name="layout">
     <bean class="java.awt.BorderLayout"/>
   </property>

   <add>
     <bean class="java.awt.Panel" id="panel">
     <add>
      <bean class="java.awt.Button" id ="view">
       <property name="label" value="View"/>
      </bean>
     </add>
    </bean>
   </add>
  </bean>
 </add>
```

A similar action could be taken by the user profile, for example, if the user intended to participate in only part of the conference. Similarly, a terminal profile could modify the BML script if there was insufficient processing power to process both audio and video media streams.

The construction of the client application might be influenced by the availability of the components. This may be done, for example, to avoid delay in downloading components across the network. If there is only an audio component available locally then the application builder may construct the application without a video component. Using the same unmodified BML script the application builder could add the video component once it had been downloaded. In another example, the BML script may specify one component for decoding, say, MPEG, but the user may have another installed. The installed component could override the suggested component according to XSL rules determined from the user profile

Whilst the present invention has been described with reference to the Internet and multicast transmissions, it will be apparent to the reader that the described modular session description and the session control system are applicable to the announcement and subsequent management of connections to media streams of a (multi)media session using other known transport mechanisms such as unicast.

Furthermore, although mechanisms for encryption, charging and other such services have not been explicitly described, it would be apparent to the reader that appropriate session descriptions and associated functions within the session control system for their processing could be readily implemented according to the mechanism required.

CLAIMS

1. A method of announcing a description of a media session, the method comprising the steps of:-

generating a session description comprising media oriented data necessary for a user to receive at least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session; and,

announcing the media session by making the session description available to potential recipients of the media session.

2. A method according to claim 1 wherein said media oriented data prescribes a number of application program components to be used in order to build an application to participate in the media session.

3. A method according to claim 2 wherein the media oriented data prescribes a manner in which the components are to be configured to build the application program

4. A method according to any preceding claim wherein the session description is generated using a structured data format.

5. A method according to claim 4 wherein the structured data format conforms to the format of Extensible Mark-Up Language.

6. A method according to any preceding claim wherein the user oriented data for the or each media stream is generated in a one or more respective media modules within the session description, and said method comprises the further steps of:-

generating a base module comprising user oriented data relevant to the media session;

providing a link between the base module and the or each media module;

announcing the media session by making the base module available to said potential recipients;

wherein the link to the or each media module permits a user to access the or each media module subsequent to the base module.

7. A method of configuring a platform for receiving a media session, said method comprising the steps of:-

receiving a session description of a media session, said session description comprising media oriented data necessary for a platform to receive at least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session;

processing said session description to determine an appropriate application program configuration form a list of available application programs or program components;

configuring a respective media session application program from said list of available programs for participation in said media session.

8. A method according to claim 7 wherein said media oriented data prescribes a number of program components to be used, and wherein said step of processing said session description comprises the step of selecting said prescribed components from said list.

9. A method according to claim 8 wherein the media oriented data prescribes a manner in which the components are to be configured, and the step of configuring said respective media session application program comprises the step of configuring said selected components according to said prescribed manner.

10. A method according to any one of claims 7 to 9 further comprising the steps of receiving network data relating to characteristics of the network over which said media session is to be transmitted and wherein the respective media session application is configured according so said network data.

11. A method according to any one of claims 7 to 10 further comprising the steps of receiving terminal data relating to characteristics of the terminal on which said media session is to be received and wherein the respective media session application is configured according so said terminal data.

12. A method according to claim 10 or claim 11 wherein said network data or terminal data or both is monitored during the media session and the media session description is modified in response to changes to the monitored data.

13. A method according to any one of claims 7 to 12 further comprising the steps

of receiving user profile data relating to preferences of a user of the media session and wherein the respective media session application is configured according so said user profile data.

14. A method according to any one of claims 7 to 13 wherein the session description further comprises data defining a quality of service policy for receiving the media session and the respective media session application is configured according to said quality of service policy.

15. A method according to any one of claims 7 to 14 wherein the session description further comprises data defining one or more remote services necessary for participation in said media session and the respective media session application is configured according to requirements of said one or more remote services.

16. A method according to any one of claims 7 to 15 at least the session description is generated using a structured data format.

17. A method according to claim 13 wherein the structured data format conforms to the format of Extensible Mark-Up Language.

18. A method according to any one of claims 7 to 17 wherein the step of processing the session description comprises the step of parsing the session description using a terminal session control to determine an appropriate application program configuration form a list of available application programs or program components; selecting one or more media streams identified in the session description; and connecting the or each selected media stream to one or more application programs or components in said

configuration by means of a session control configured for managing media stream connections for the or each application program or component.

19. A system for announcing a description of a media session, the system comprising:-

a media session description generator for generating a session description comprising media oriented data necessary for a user to receive at least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session; and,

a media session announcer for announcing the media session by making the session description available to potential recipients of the media session.

20. A system for configuring a platform for receiving a media session, said system comprising:-

a receiver for receiving a session description of a media session, said session description comprising media oriented data necessary for a platform to receive at-least one media stream of a media session, said media oriented data identifying one or more application program components or requirements for one or more application programs or configurations of application program components necessary to participate in said media session;

a processor for processing said session description to determine an appropriate application program configuration form a list of available application programs or program components; and,

an application builder for configuring a respective media session application

program from said list of available programs for participation in said media session.

21. A computer readable storage medium containing executable instructions for performing the methods of any one of claims 1 to 18.

22. A computer readable storage medium containing the system according to claim 19 or claim 20.

ABSTRACT

SESSION ANNOUNCEMENT FOR ADAPTIVE COMPONENT CONFIGURATION

The invention provides a method and system for announcing a description of a media session such as a multimedia conference that is to take place over a multicast capable network, for example the communications network known as the MBone (Internet Protocol Multicast Backbone). The invention also provides a method and system for configuring a platform for receiving such a media session. The method of announcing a session description comprises the steps of:- generating a session description (1115) comprising media oriented data necessary for a user to receive at least one media stream of a media session, wherein the media oriented data identifies one or more application program components (1145) or requirements for one or more application programs or configurations of application program components (1145) necessary to participate in the media session; and, the step of announcing the media session by making the session description available to potential recipients (1135) of the media session. In another aspect the of invention the method of configuring a platform for the media session comprises the steps of:- receiving a session description (1115) of the aforementioned type and processing the session description to determine an appropriate application program configuration (1150) form the list (1140) of available application programs or program components (1145); and, the step of configuring a respective media session application program (1155) from the list (1140) for participation in said media session.

Figure 11

Fig.1.

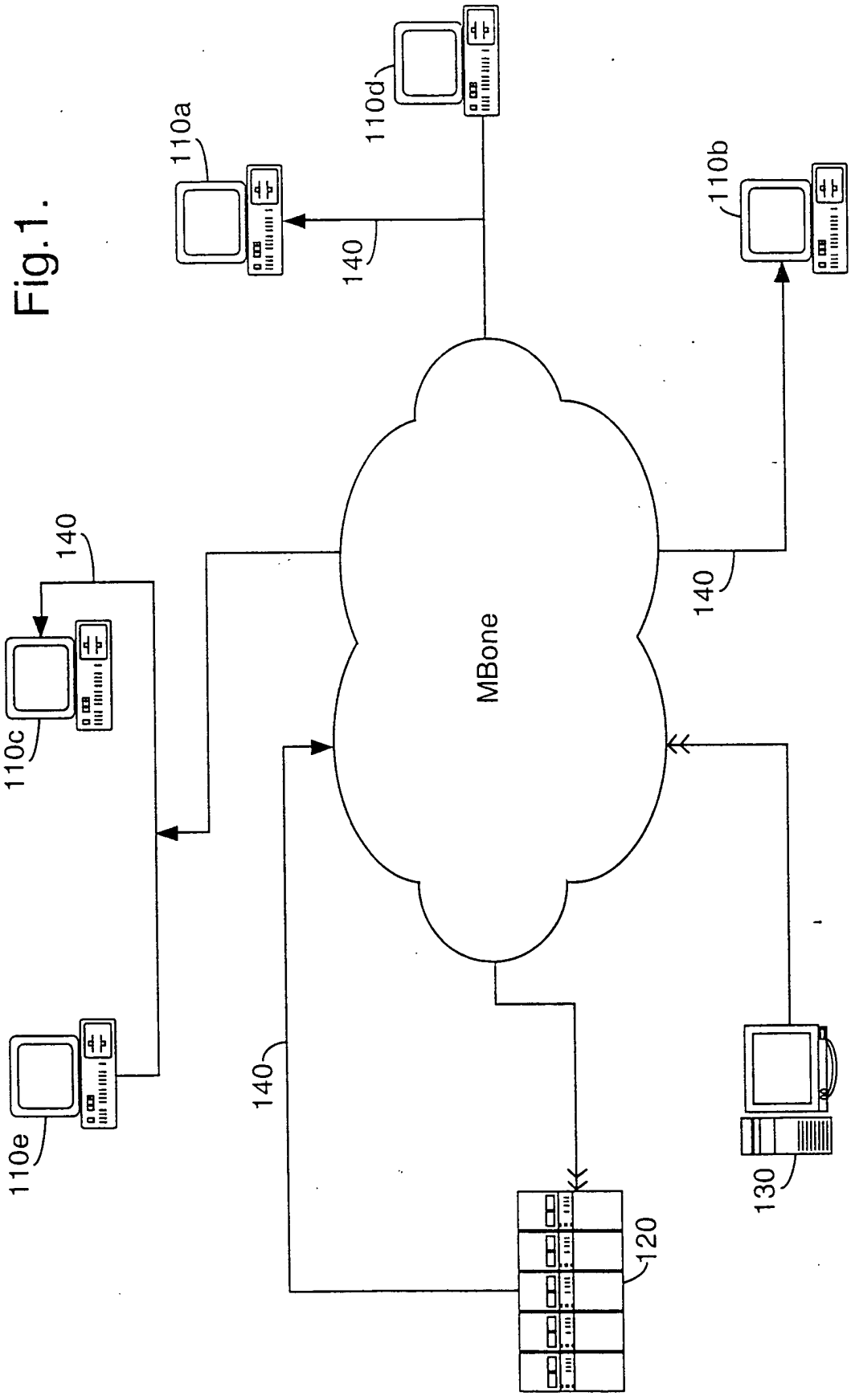


Fig.2.

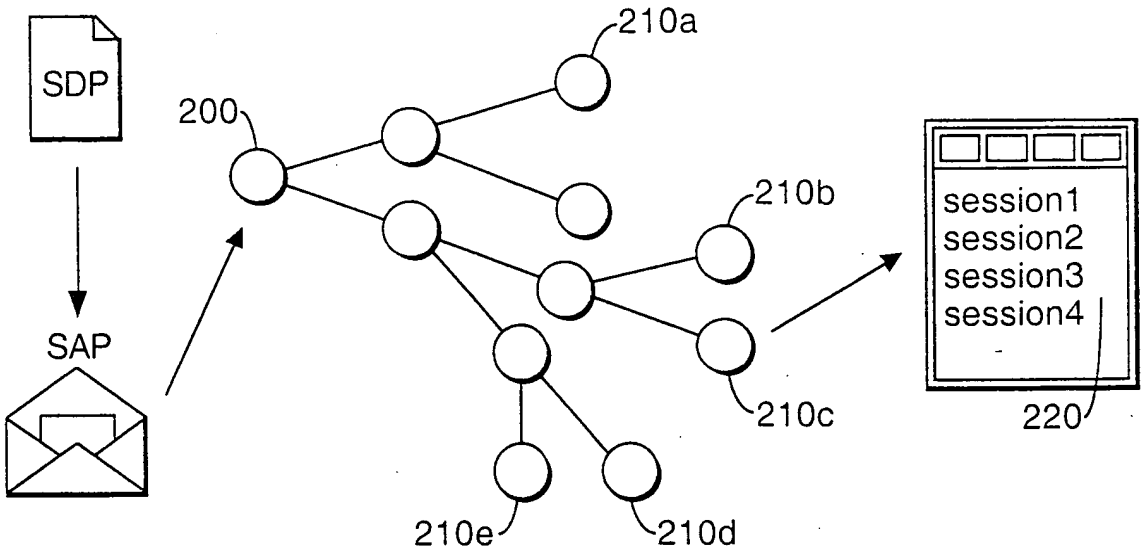

SDP
SAP
200
210a
210b
210c
210d
210e
session1
session2
session3
session4
220

Fig.3.

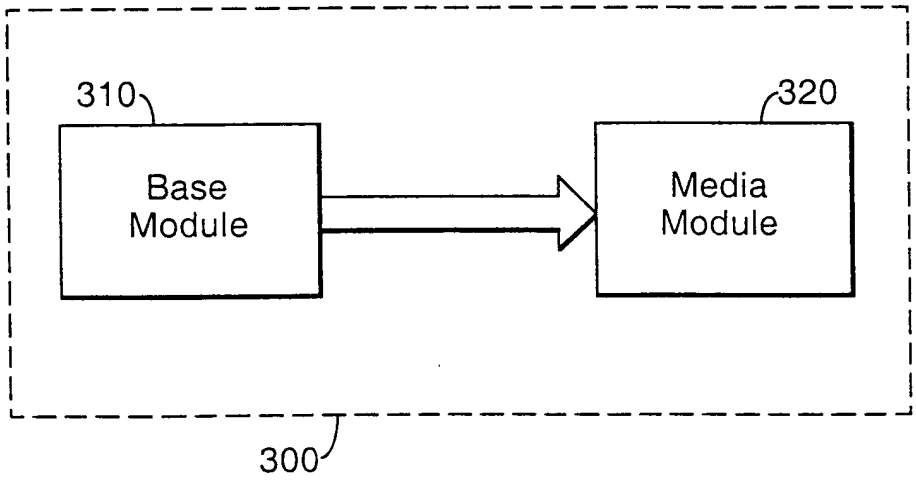

310
Base Module
320
Media Module
300

Fig.4.

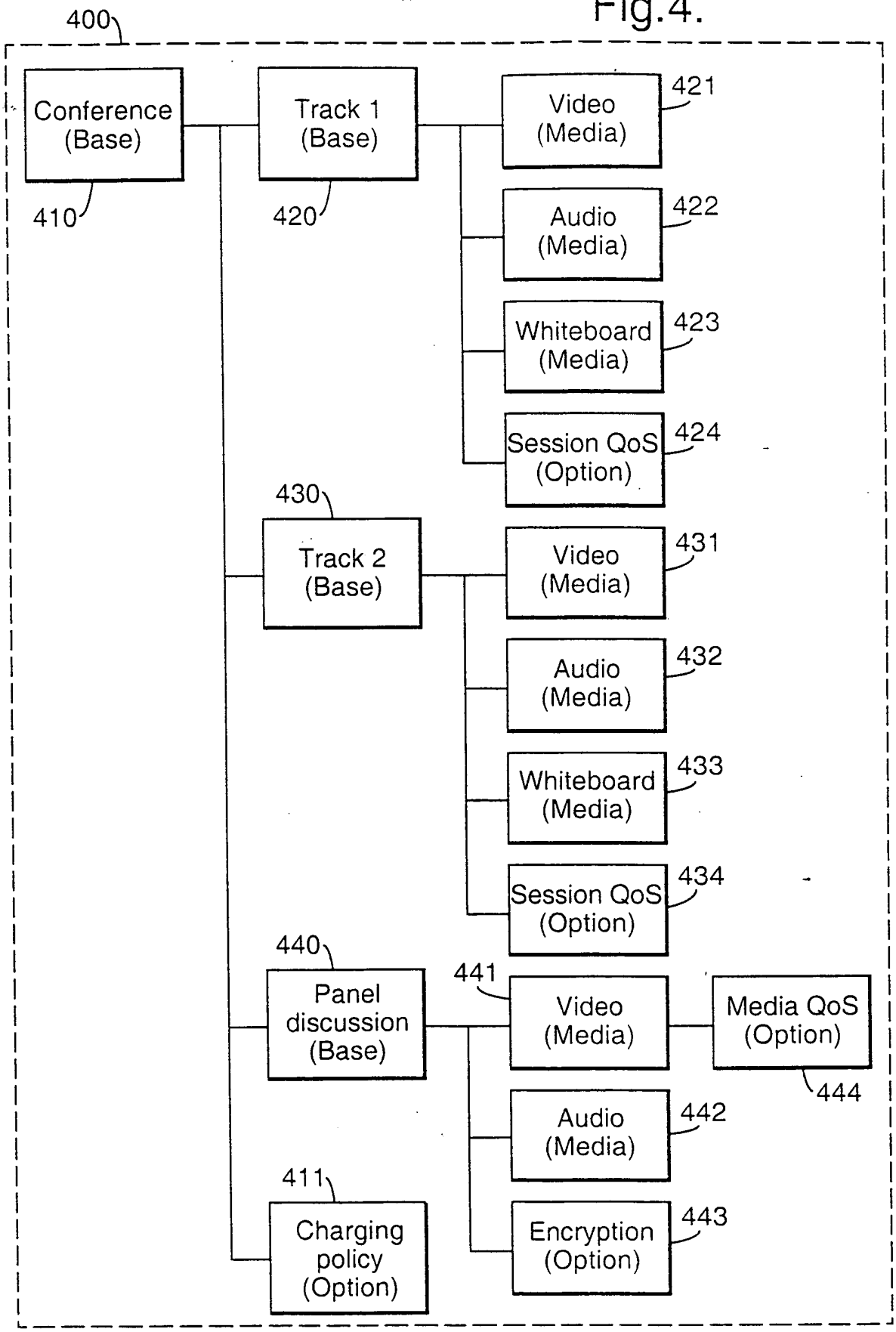

400
Conference (Base)
410
Track 1 (Base)
420
Video (Media)
421
Audio (Media)
422
Whiteboard (Media)
423
Session QoS (Option)
424
430
Track 2 (Base)
Video (Media)
431
Audio (Media)
432
Whiteboard (Media)
433
Session QoS (Option)
434
440
Panel discussion (Base)
441
Video (Media)
Media QoS (Option)
444
Audio (Media)
442
411
Charging policy (Option)
Encryption (Option)
443

Fig.5.

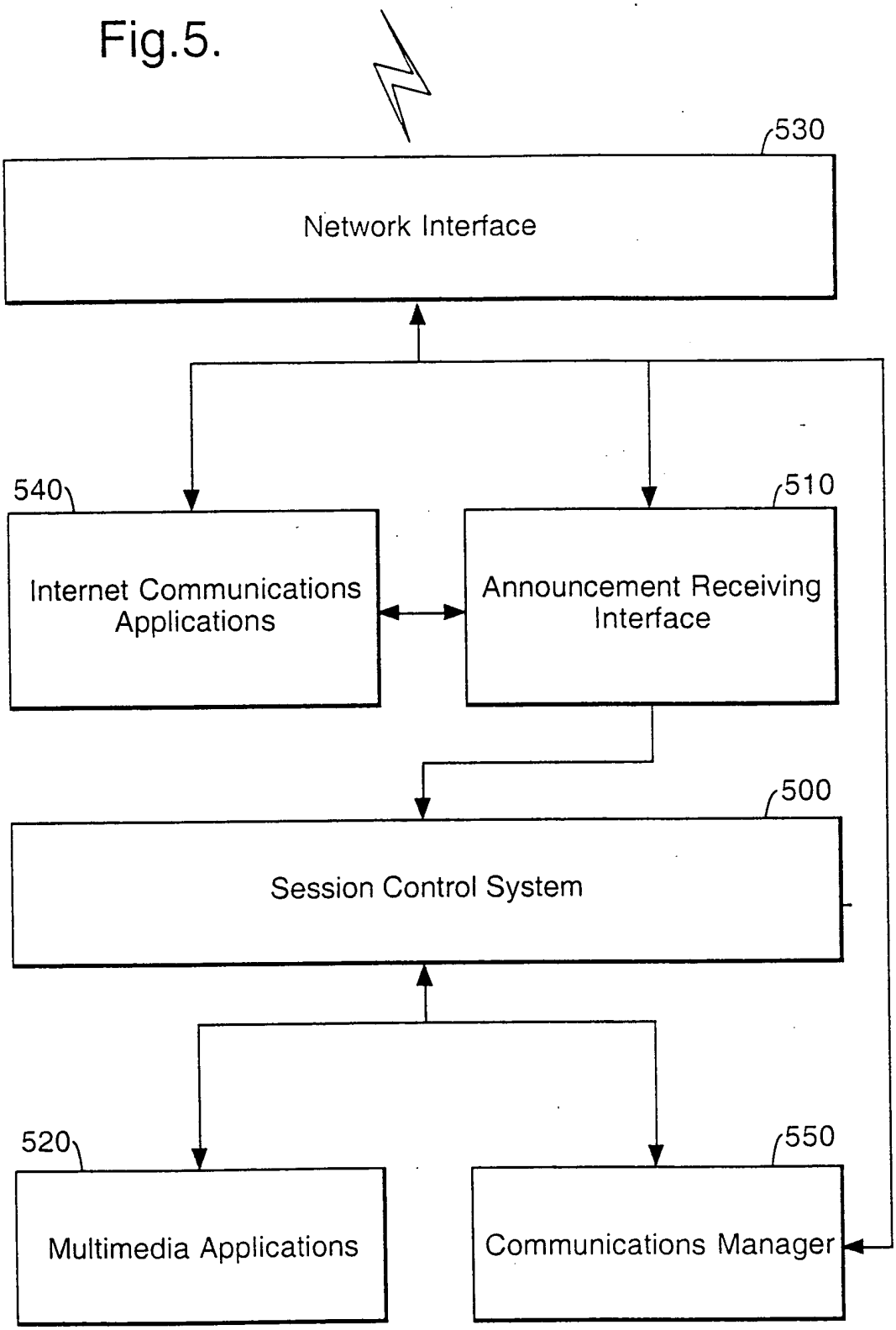

530
Network Interface
540
Internet Communications Applications
510
Announcement Receiving Interface
500
Session Control System
520
Multimedia Applications
550
Communications Manager

Fig.6.

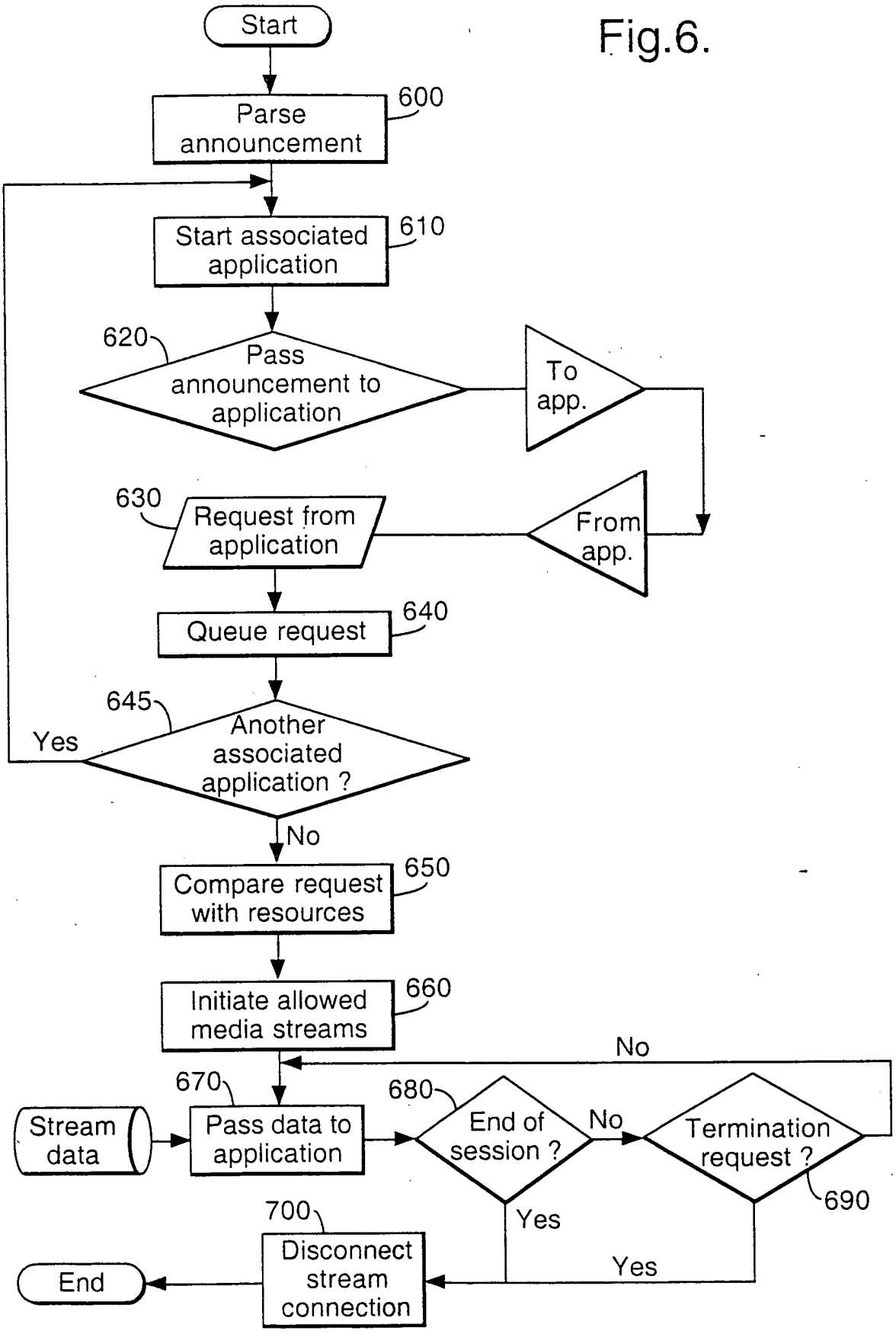

Start
Parse announcement
600
Start associated application
610
620
Pass announcement to application
To app.
630
Request from application
From app.
Queue request
640
645
Another associated application ?
Yes
No
Compare request with resources
650
Initiate allowed media streams
660
No
670
680
Stream data
Pass data to application
End of session ?
No
Termination request ?
690
Yes
700
Yes
Disconnect stream connection
End

Fig.7.

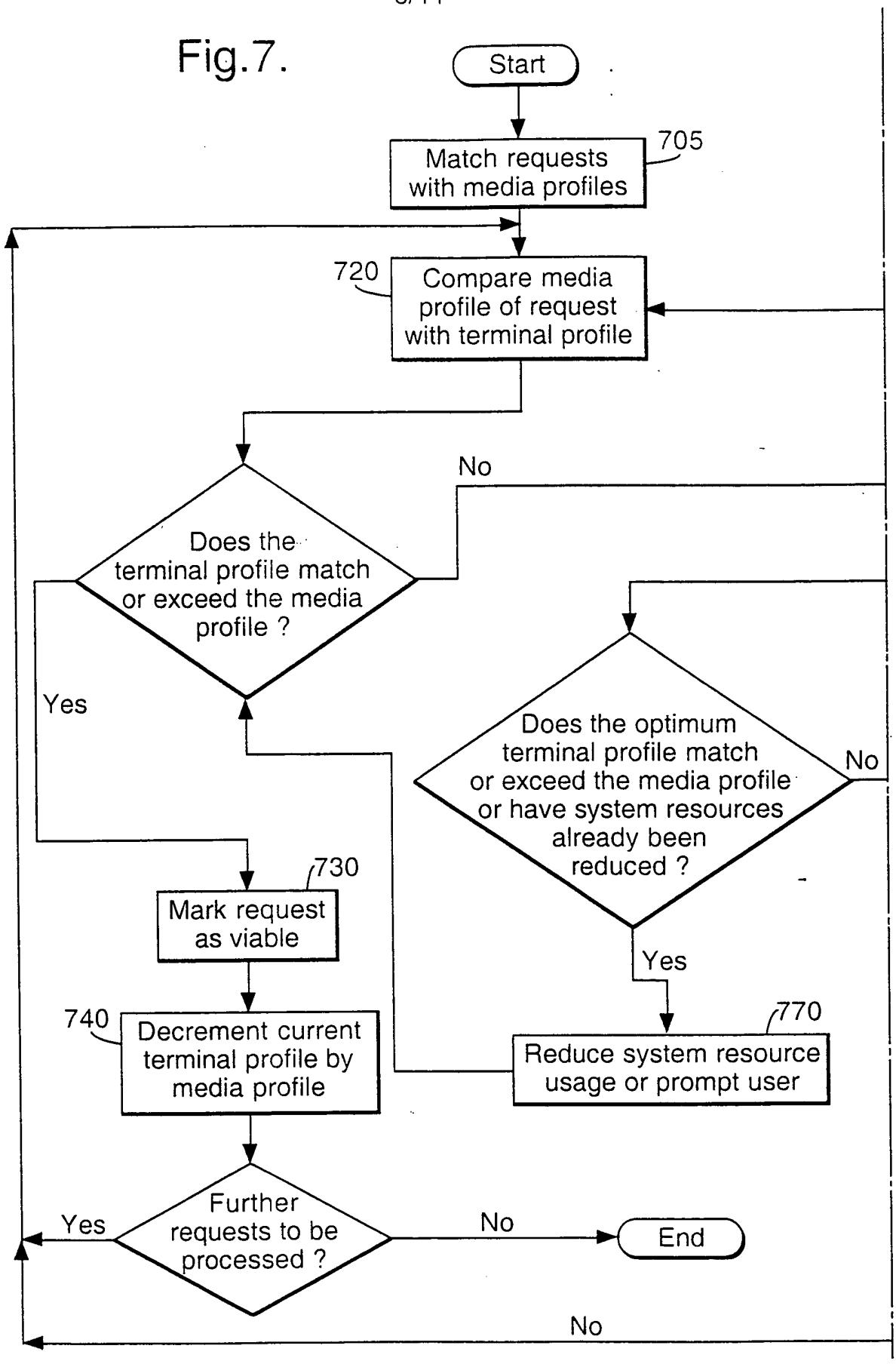

Start
Match requests with media profiles
705
720
Compare media profile of request with terminal profile
Does the terminal profile match or exceed the media profile ?
No
Yes
Does the optimum terminal profile match or exceed the media profile or have system resources already been reduced ?
No
Yes
730
Mark request as viable
740
Decrement current terminal profile by media profile
770
Reduce system resource usage or prompt user
Further requests to be processed ?
Yes
No
End
No

Fig.7 (Cont).

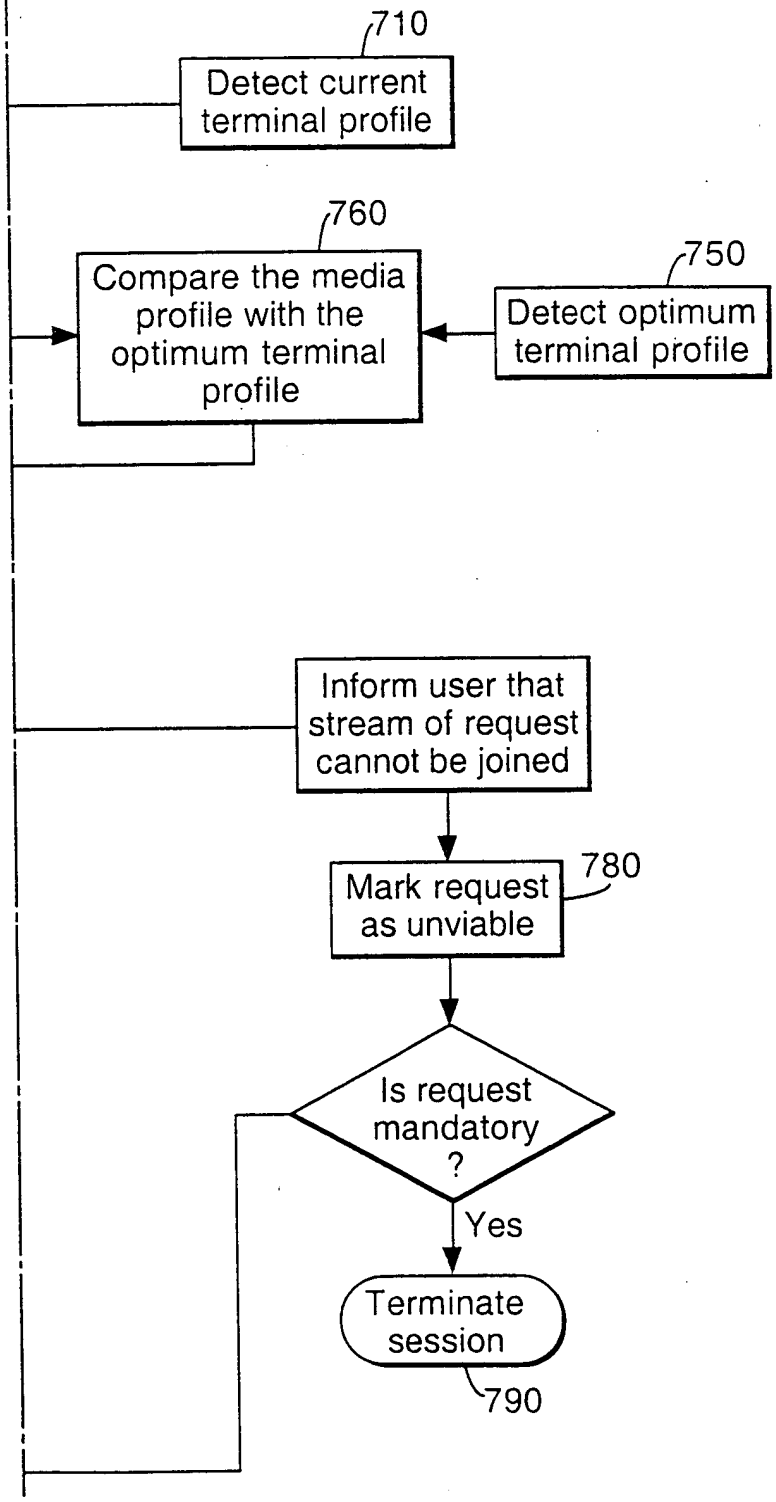

710
Detect current terminal profile
760
Compare the media profile with the optimum terminal profile
750
Detect optimum terminal profile
Inform user that stream of request cannot be joined
780
Mark request as unviable
Is request mandatory ?
Yes
Terminate session
790

Fig.8.

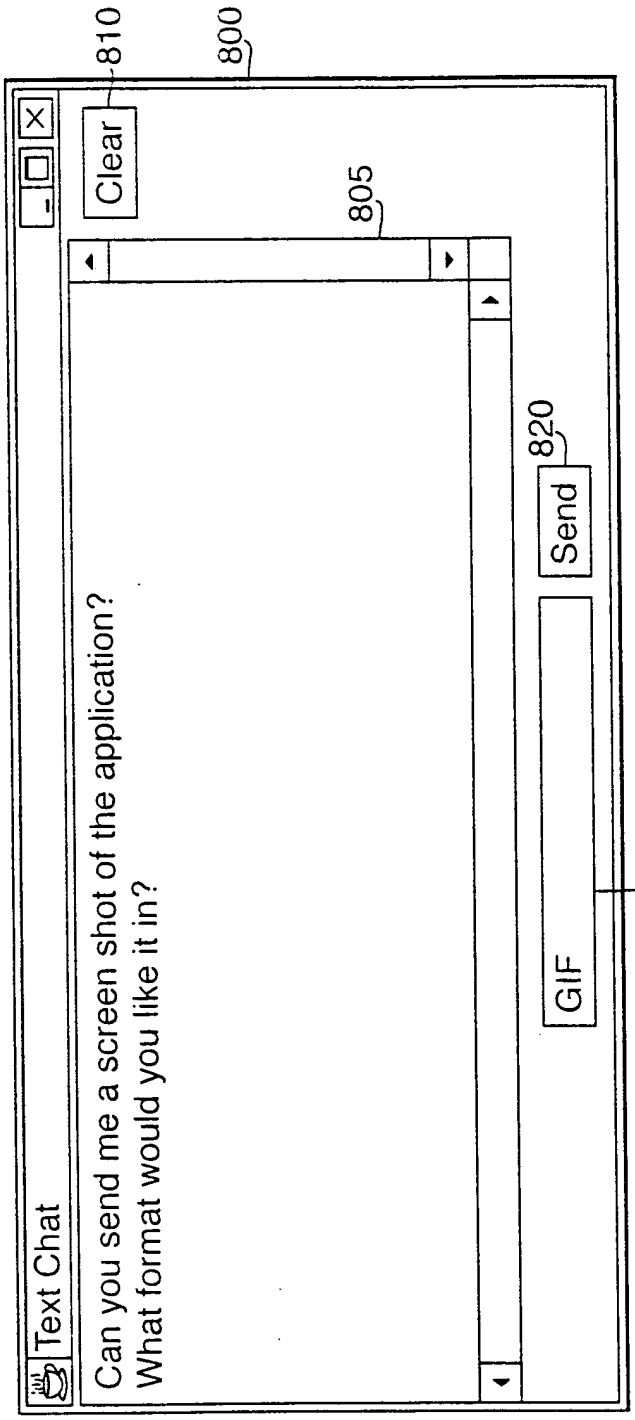

Text Chat
Clear
Can you send me a screen shot of the application?
What format would you like it in?
Send
GIF
800
805
810
815
820

Fig.9.

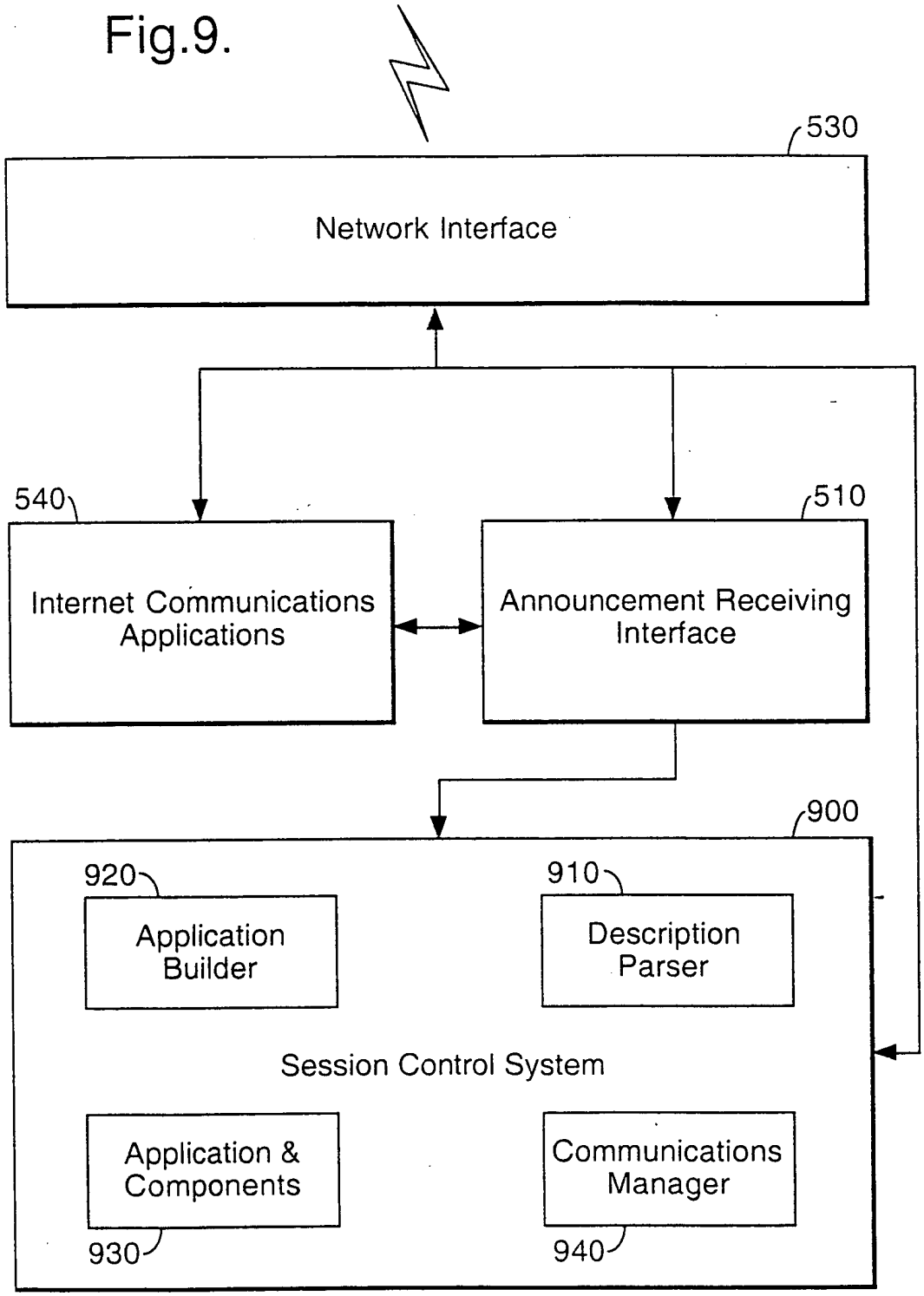

530
Network Interface
540
Internet Communications Applications
510
Announcement Receiving Interface
900
920
Application Builder
910
Description Parser
Session Control System
Application & Components
930
Communications Manager
940

Fig.10.

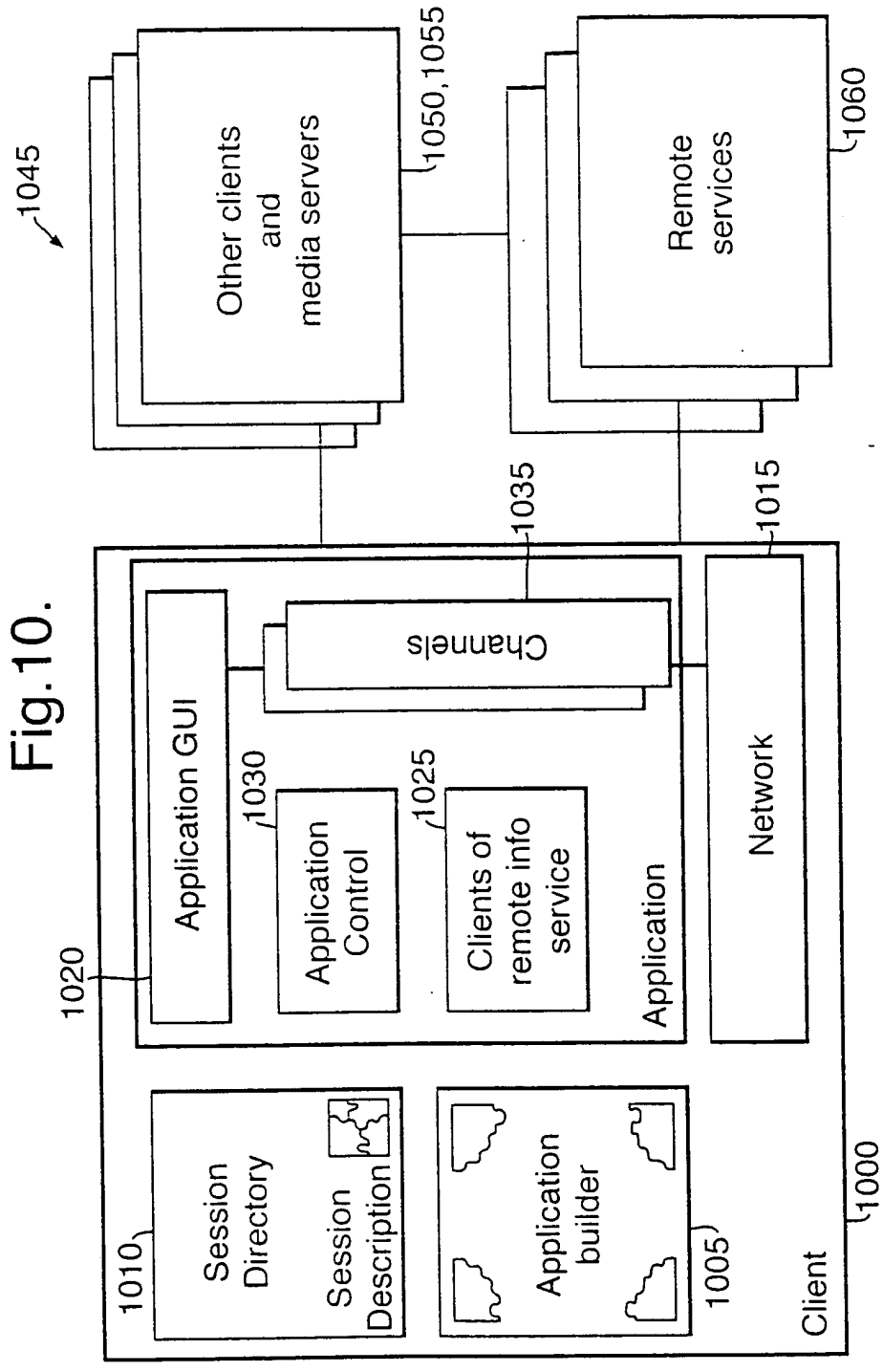
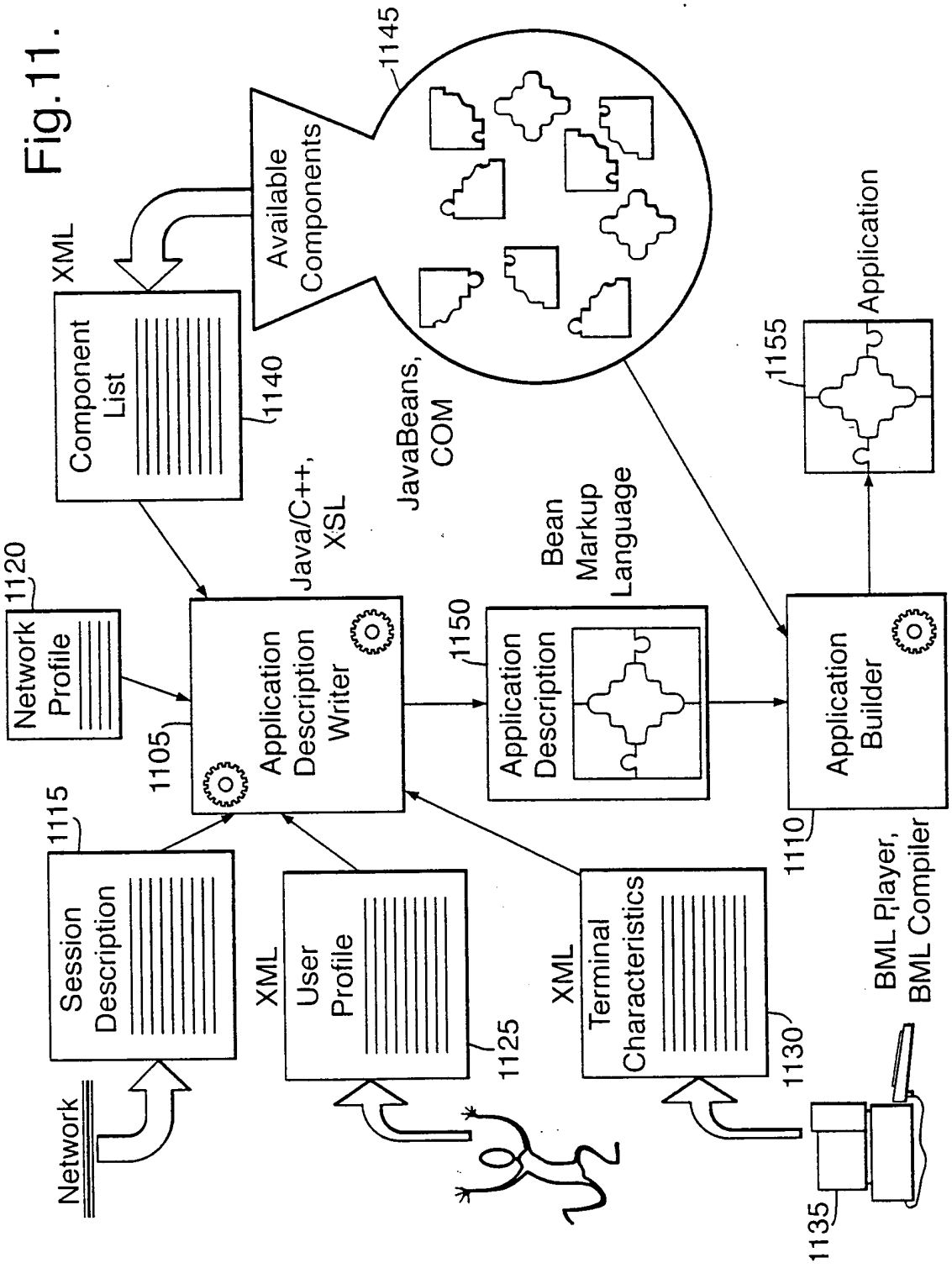

1045
Other clients and media servers
1050,1055
Remote services
1060
1035
1015
Channels
Application GUI
1020
1030
Application Control
1025
Clients of remote info service
Application
Network
Session Directory
Session Description
1010
Application builder
1005
Client
1000

Fig.11.


Available Components
1145
XML
Component List
1140
JavaBeans, COM
Java/C++, XSL
Bean Markup Language
Application
1155
Network Profile
1120
1105
Application Description Writer
1150
Application Description
Application Builder
1110
Session Description
1115
XML
User Profile
1125
XML
Terminal Characteristics
1130
BML Player, BML Compiler
Network
1135